Milestone 1 Report

Name	ID
Adel Mahmoud Mohamed Abdelrahman	20010769
Mohamed Hassan Sadek Abdelhamid	20011539
Mohamed Hassan Yousef Hassan Eltobgy	20011544
Mohamed Raffeek Mohamed El-Sayed	20011574
Mahmoud Attia Mohamed Abdelaziz Zian	20011810
Mahmoud Ali Ahmed Ali Ghallab	20011811

Requirements

1.1: Customer Sign-Up Using Basic Credentials

Function: Allow the customer to create a new account using basic credentials (email and password).

Description:

- Customers can sign up using basic credentials (email and password) to create a new account.
- The sign-up form for the user will contain **the following fields**:
  - user email
  - password
    - the user email should be in the following format ([username]@[domain]) and have a minimum length of 6 characters and max length of 40 characters.
    - the password should have minimum length of 8 characters and maximum length of 20 characters and consists of the following:
      - English letters (uppercase and lowercase): A-Z, a-z
      - Numbers: 0-9
      - Symbols: Some of the symbols (! @ # $ % ^ & * ( ) _ + - { } [ ] | \ : ; " ' < > , . ? /)
    - If there's any incorrect format in either of the two fields, the user will receive a notification through a red error message displayed around the field containing the error on the frontend form. In such cases, the data cannot be submitted to the backend.
- **At the end of the form there are 3 buttons**:
  - **Sign up:**
    - to send this data to the backend to complete the sign-up process.
  - **Sign-up using Gmail:**
    - to redirect the customer to Google's authenticator service.
  - **Already have an account? login.**
    - to navigate to the login form

Inputs: Email and Password

Source: The sign-up form.

Outputs:

- In case of successful sign up:
  - send a verification email to the user with a link to navigate to home page when clicking.
  - notify the user to open the sent verification email.
- In case of failure sign up (ex: user already exist):
  - notify the user with the failure response to try sign up again with valid info.

Destination:

- In case of successful sign up: pop-up window in the sign-up form, and the provided email (an email is sent there).
- In case of failure sign up (ex: user already exist): pop-up window in the sign-up form.

Action:

- In the backend handle the sign-up request called from the front which will include the user email and password.
- In case of valid sign up a verification email with a link will be sent to the user email and response will be sent to the front to tell the user to verify through his email.
- when the user clicks the link provided in his email, should be navigated to the frontend home page.
- In case of invalid sign up (user already exist as email can't be repeated) return the failure response message to the frontend.

Requirements: Email, password, a click on the sign-up button.

Pre-condition: The backend server is up and running, the user has a genuine email address.

Post-condition: None.

Side effects: The user is added to the database on successful sign-up.

1.2: Customer Sign-Up Using Gmail
Function: Allow the customer to create a new account using Google's authenticator services.

Description:
• Customers can sign up using their Gmail to create a new account.
• Here we will use the Sign-Up using Gmail button in the sign-up form:
  ◦ **Sign-up using Gmail:**
  ▪ to redirect the customer to Google's authenticator service.

Inputs: Gmail
Source: The sign-up form.

Outputs:
• In case of successful sign up:
  ◦ notify the user with the successful sign up and navigate to the home page
• In case of failure sign up (ex: user already exist):
  ◦ notify the user with the failure response to try sign up again with valid info.

Destination: • In case of successful sign up: pop-up window in the home page. • In case of failure sign up (ex: user already exist): pop-up window in the sign-up form.
Action:

- In the backend handle the register post request called from the front which will include the user Gmail token.
- In case the user does not exist, it's handled as Gmail sign up, which is the case here.
- In case of already exist so it's handled as Gmail login.

Requirements: Gmail account.

Pre-condition: The backend server is up and running, the user has a genuine email address.

Post-condition: None.

Side effects: The user is added to the database on successful sign-up.

2.1: Customer Login Using Basic Credentials

Function: Allow the customer to login into his account using basic credentials (email and password).

Description:

- Users can log in to their account using basic credentials (email and password).
- The sign-in form for the user will contain **the following fields:**
  - **user email**
  - **password**
    - the user email should be in the following format ([username]@[domain]) and have a minimum length of 6 characters and max length of 40 characters.
    - the password should have minimum length of 8 characters and maximum length of 20 characters and consists of the following:
      - English letters (uppercase and lowercase): A-Z, a-z
      - Numbers: 0-9
      - Symbols: Some of the symbols (! @ # $ % ^ & * ( ) _ + - { } [ ] | \ : ; " ' < > , . ? /)
    - If there's any incorrect format in either of the two fields, the user will receive a notification through a red error message displayed around the field containing the error on the frontend form. In such cases, the data cannot be submitted to the backend.
- **At the end of the form there is 3 buttons:**
  - **Sign in:**
    - to send this data to the backend to complete the sign-in process.
  - **Sign-in using Gmail:**
    - to redirect the customer to Google's authenticator service.
  - **Don't have an account? Sign Up.**
    - to navigate to the Sign-Up form

Inputs: Email and Password

Source: The login form.

Outputs:

- In case of **successful** log in:
  - notify the user with the successful log in and navigate to the home page.
- In case of **failure** log in (ex: user not found or incorrect password):
  - notify the user with the failure response to try log in again with valid info.

Destination:

- In case of successful login: pop-up window in the home page.

- In case of failure login: pop-up window in the login form.

Action:

- In the backend handle the sign-in request called from the front which will include the user email and password.
- In case of successful sign in a successful response will be sent to the front to notify the user before navigating to home page.
- In case of invalid sign in (the user is not existing or incorrect password) return the suitable failure message to the frontend to notify the user.

Requirements: Email, password, a click on the login button.

Pre-condition: The backend server is up and running, the user has a registered email address and its correct password.

Post-condition: None.

Side effects: None.

2.2: Customer Login Using Gmail

Function: Allow the customer to login using Google's authenticator services.

Description:

- Customers can login using their Gmail to access the website.
- Here we will use the Login using Gmail button in the login form:
  - **Sign-in using Gmail:**
    - to redirect the customer to Google's authenticator service.

Inputs: Gmail

Source: The login form.

Outputs:

- In case of successful login:
  - notify the user with the successful login and navigate to the home page.
- In case of failure login:
  - notify the user with the failure response to try login again with valid info.

Destination:

- In case of successful login: pop-up window in the home page.
- In case of failure login: pop-up window in the login form.

Action:

- In the backend handle the login post request called from the front which will include the user Gmail token.
- In case the user does not exist, it's handled as Gmail sign up.
- In case of already exist so it's handled as Gmail login, which is the case here.

Requirements: Gmail account.

Pre-condition: The backend server is up and running, the user has a genuine email address.

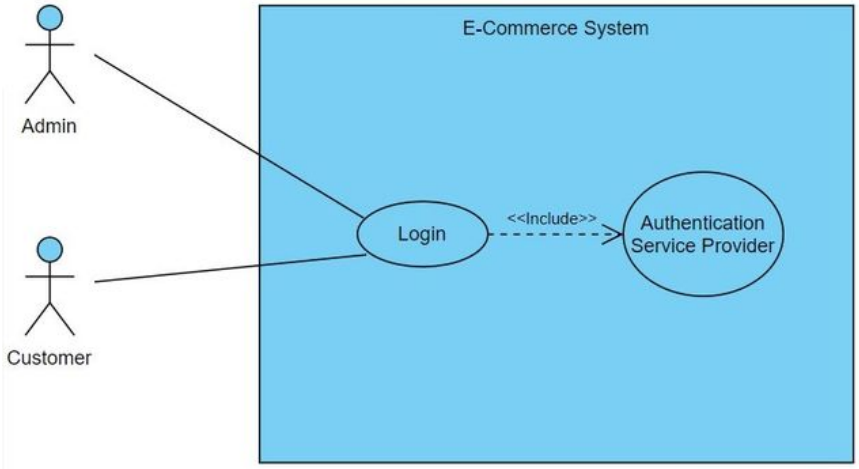
Post-condition: None.

Side effects: None.

Diagrams

Login Use Case Diagram

Login Usecase Diagram.pdf

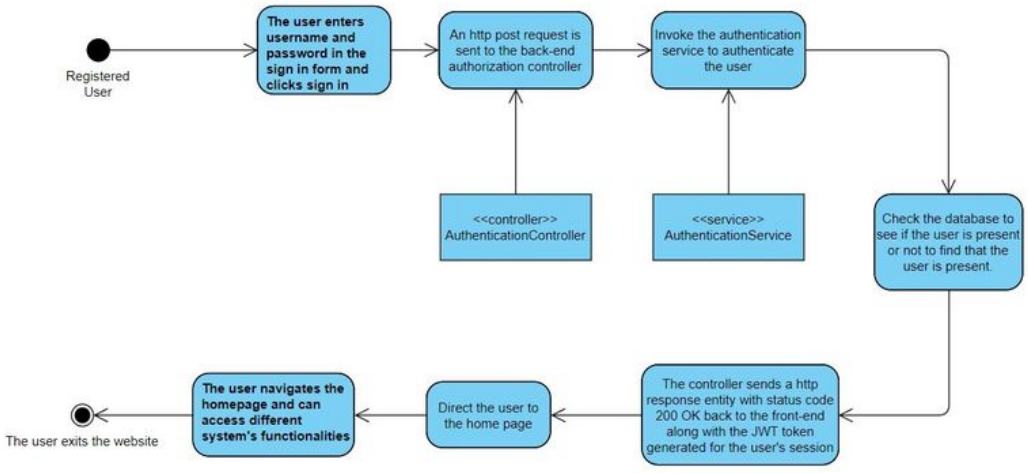


The login feature enables users to securely access the E-Commerce system by utilizing their basic credentials. Through the invocation of relevant controllers and the provision of authentication services, users undergo a secure authorization process, ensuring their legitimate access to the system.

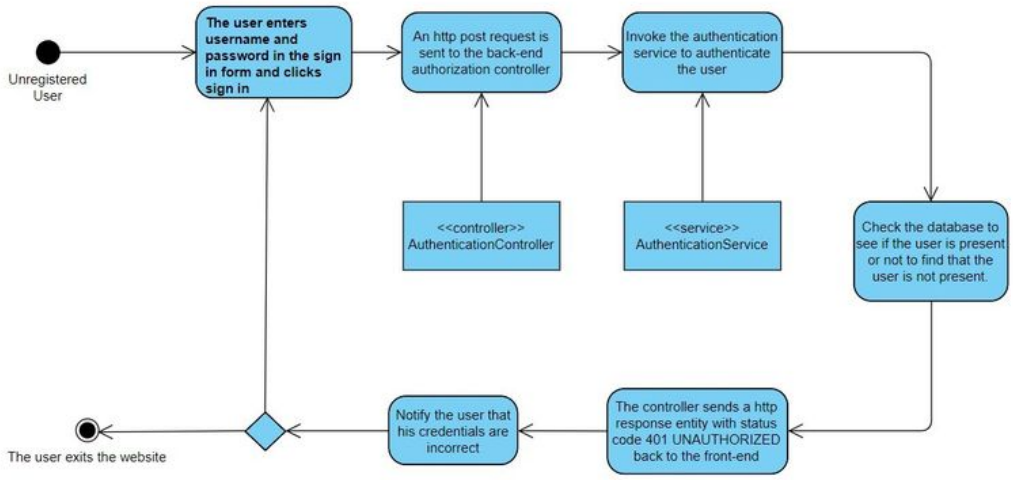
Login Activity Diagrams

Login Activity Diagrams.pdf

Authorized user:

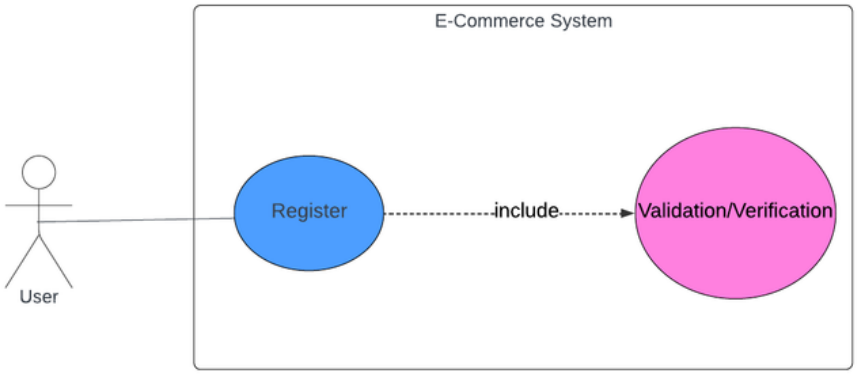


Unauthorized user:



Register Use Case Diagram

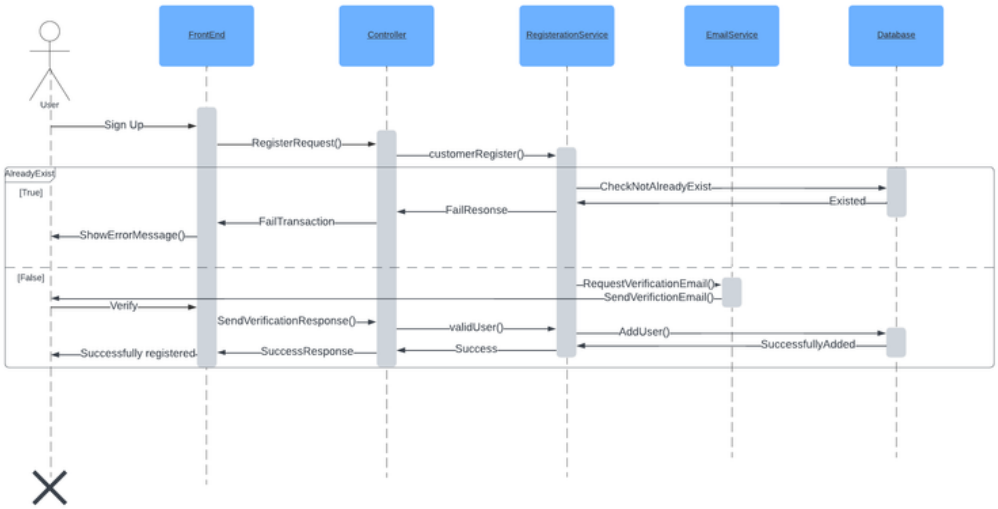
Register Use case diagram.png



Note: Register functionality allows user to create new account in the system it includes the required information such as his Email address and Password. the function also includes validation logic to make sure that the user who trying to sign up to the system is a new user (i.e not existent one) by checking the Email address -he already entered- is already existed or not in order to keep the Email adresses unique to everyone.
Register functionality also include the Verification logic in which the system will send a verification email to the user and not allow user to be registered till he verified by clicking on that verification email. if he checked it correctly, then the system will add this user to the system and he can log in with the information he already filled.

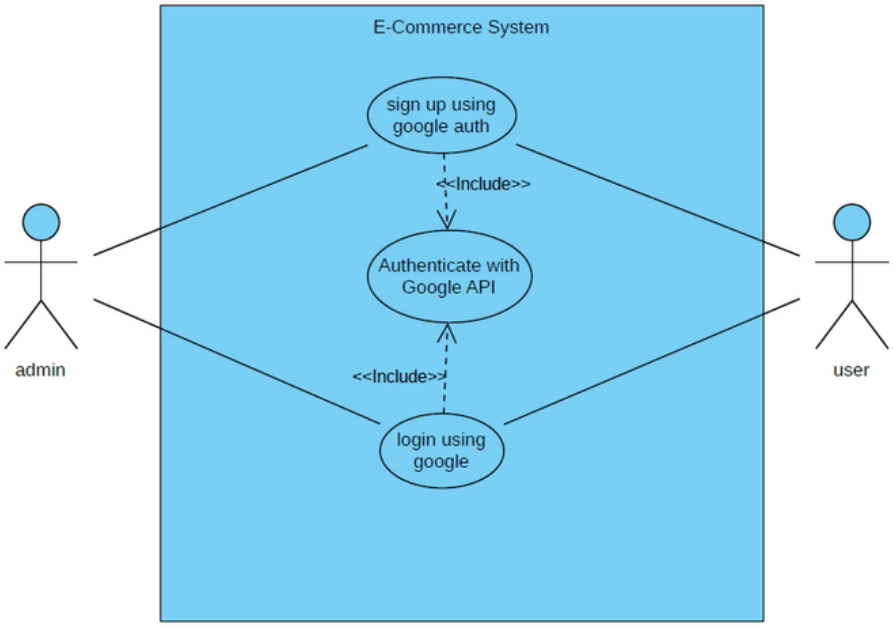
Register Sequence Diagram

Rrgister Sequence diagram.png



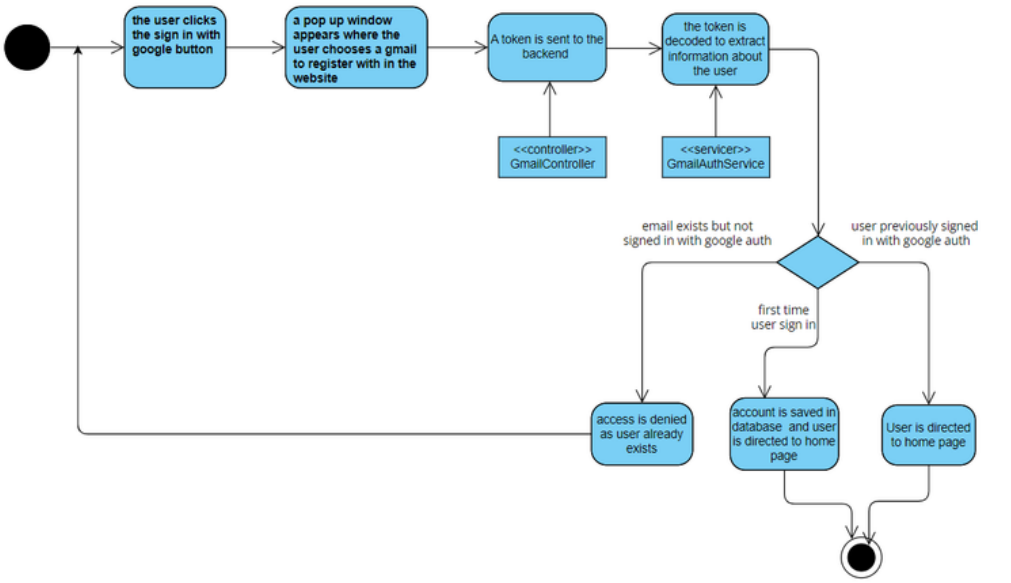
Google Authentication Use Case Diagram

use case google.pdf



Google Authentication Activity Diagram

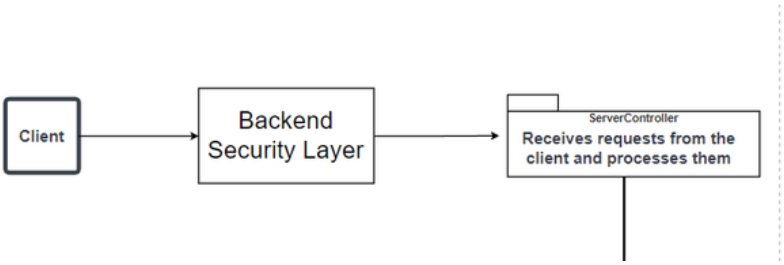
activity google auth.pdf



Design Changes

A new security layer has been added to our system

The class diagram is now modified to include this layer as follows:

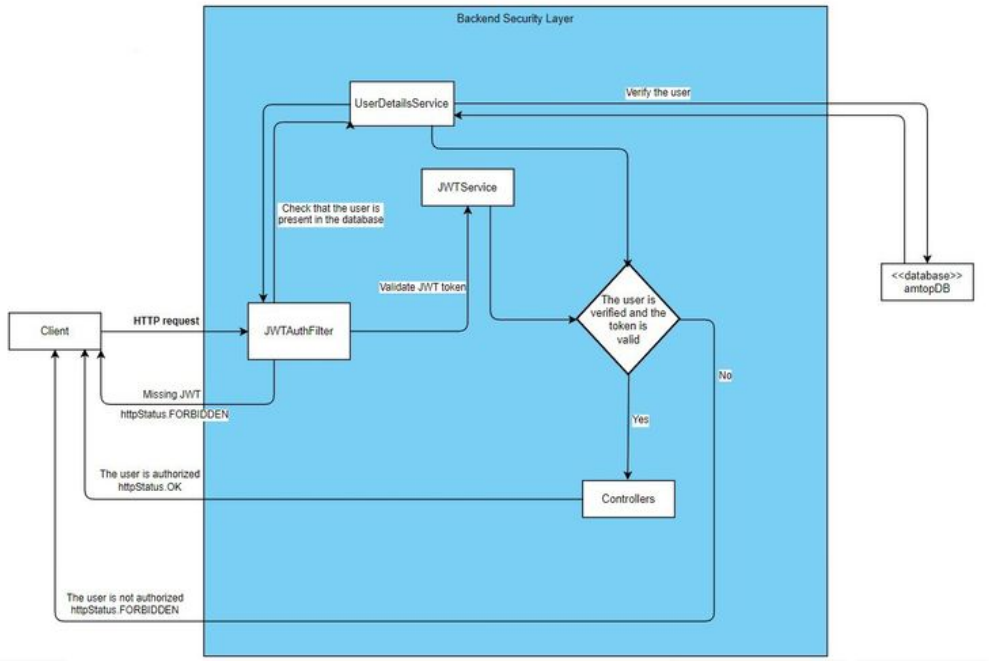


The security layer is decomposed into the following modules:

- JWTAuthFilter
- JWTService
- UserDetailsService
- SecurityConfigurations

See the following diagram to clarify how the requests are intercepted by the security layer.

Security flow.pdf

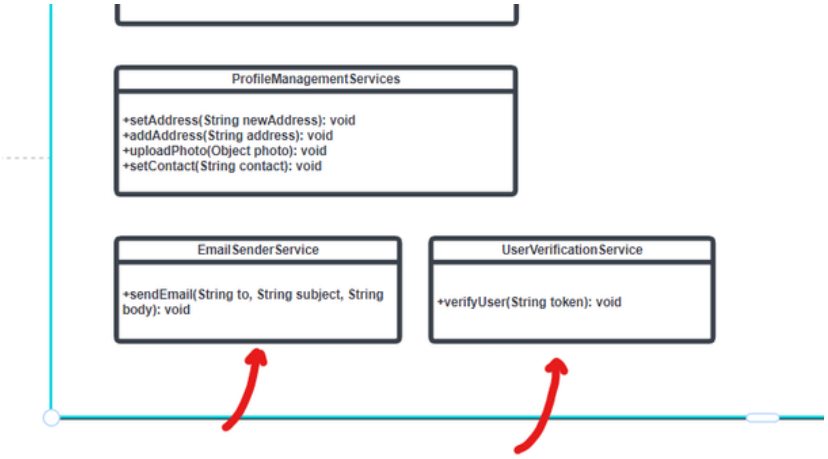


The flow is as follows:

- The client sends a request to the backend server.
- The request is intercepted by the JWTAuthFilter.
- The filter needs to know two things:
  - the user exists in the database.
  - the sent JWT token is valid.
- The filter calls the needed modules to evaluate these conditions.
- If both conditions are satisfied, the request can finally reach the rest controllers, otherwise, access to the backend server is denied.
- **Note that** there are two controllers that can be accessed directly (whitelisted), which are the authentication controller and the google authentication controller. This is specified according to our design decisions.
- **Also note that** the JWT token mentioned above is generated on the user login or user successful email validation, and this token is then passed to the frontend to be used by the user in case he wanted to access any of the backend functionality. However, this token is not permanent, and has an expiration date of 1 day, therefore a user can have a valid session for 24 hours before having to re-login.

Customer Verification

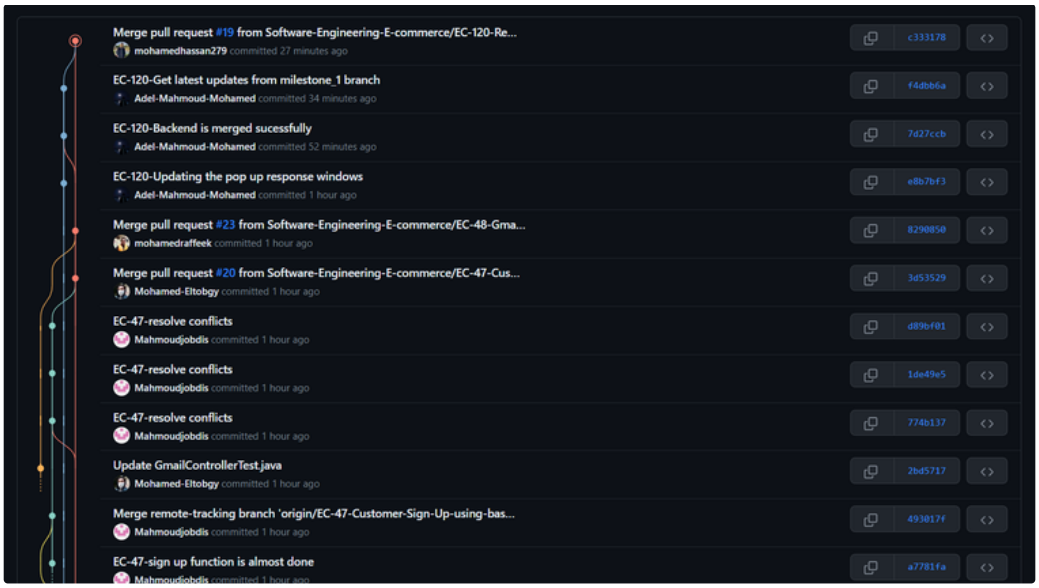
Two services were added:

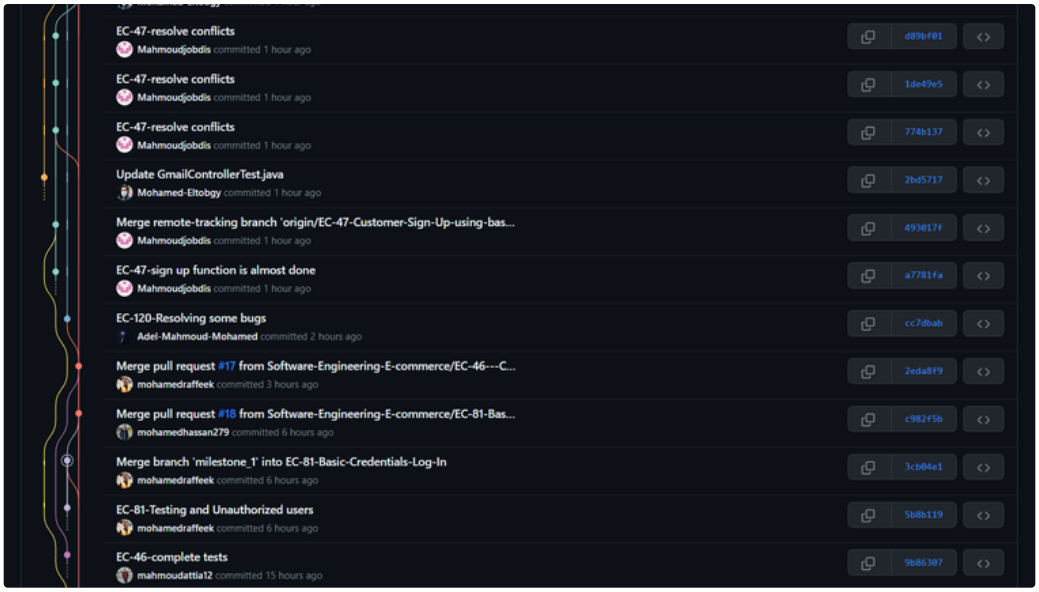


They are used as follows:

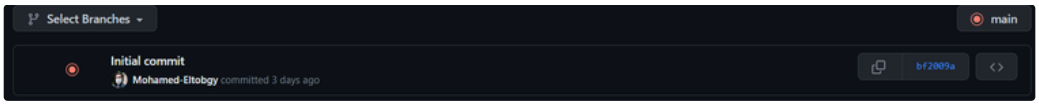
- When a new customer is registered, the EmailSenderService sends a verification email to his email address.
- When the customer goes into his email and clicks on the verification button, the UserVerificationService is invoked, which verifies the user in the database.

Git Branching Model





Note that the branching model is huge, we can check it fully during the discussion since it wont fit here.



It can be shown here that the main branch is empty completely and not a single commit has been made on it.

Code

Software-Engineering-E-commerce/AMTop-Ecommerce: Software Engineering project (github.com)